REMARKS

I. Introduction

Claims 1 to 21 are currently pending in the present application. In view of the following remarks, it is respectfully submitted that all of the presently pending claims are allowable, and reconsideration of the present application is respectfully requested.

II. Rejection of Claims 1 to 21 Under 35 U.S.C. § 102(b)

Claims 1 to 21 were rejected under 35 U.S.C. § 102(b) as anticipated by Richard Mateosian, Operating System Support - The Z8000 Way, Computer Design, May 1982, at 255 ("Mateosian"). Applicant respectfully submits that Mateosian does not anticipate the present claims at least for the following reasons.

As an initial matter, Applicant notes that the Office Action alleges that it has presented a strong prima facie case of obviousness. Applicant notes that claims 1 to 21 were rejected under 35 U.S.C. § 102(b), as anticipated by Mateosian. None of claims 1 to 21 were rejected under 35 U.S.C. § 103 as obvious over Mateosian. Indeed, the Office Action has not presented any case of obviousness whatsoever.

Claim 1 recites, inter alia, the following:

> **. . . a first task control block associated with the task and located in a first area of the memory space, the first task control block including a number of first task information data structures that contain first task information;**
>
> **a second task control block associated with the task and located in a second area of the memory space, the second task control block including a number second task information data structures that contain second task information;**
>
> **wherein the first area of the memory space is not directly accessible by the task, and the second area of the memory space is directly accessible by the task.**

The Office Action asserts that figure 3 of Mateosian discloses the recited task. The Office Action alleges that figure 3 also discloses the recited first and second task control blocks associated with the task. However, Mateosian states that figure 3 only lists software components representative of system services that all operating systems must supply. Page 256, column 1, lines 35 to 37. That is,

figure 3 illustrates categories of software services to be performed and the particular services of each of those categories. Figure 3 does not illustrate ***task control blocks*** associated with a particular task, where the task control blocks include ***data structures that contain information*** pertaining to the particular task with which the task control blocks are associated.

Addressing Applicant's response filed October 14, 2003, the Office Action asserts that the listed services of figure 3 must be blocks of software on memory. While executable code of software is loaded into memory for execution of each instruction, Mateosian does not at all discuss the memory-loaded instructions, much less the task information pertaining to a task instance of the software element. Thus, for example, Mateosian does not disclose, or even suggest, that the software instructions of a particular software element, or task information of a task of the software element, are divided into two sets of data structures, the two sets located in different memory areas, each set containing task information associated with the same particular task of the software element, such that the task of the software element can access one of the memory areas and not the other.

The Office Action further alleges that the program status pointer control registers, the system mode stack registers, process manager, and system status disclose the first memory area not directly accessible by the task, and that the normal mode stack register discloses the second memory area directly accessible by the task.

As an initial matter, with respect to the process manager and system status, as discussed above, figure 3 illustrates only a list of system elements with a description of the functions thereof, not a memory area to be accessed by a task. As described in Mateosian, the list of elements of figure 3 are ***run***, not accessed, while in system mode. Page 256, column 2, lines 54 to 56. Thus, process manager and system status of figure 3 do not disclose, or even suggest, the memory area inaccessible by a task with which the task control block of the memory area is associated.

Furthermore, as discussed above, the Office Action relies upon the list of elements of figure 3 as disclosing task control blocks associated with a task. As discussed above, Applicant respectfully disagrees with this assertion. The listed services of figure 3 do not disclose, or even suggest, the first and second task control blocks associated with a particular task, where the task control blocks are

located in different memory space areas, one area directly accessible by the particular task with which the task control blocks are associated, and the other area not directly accessible by the particular task with which the task control blocks are associated. Mateosian states that the elements of figure 3 are ***all*** run in system mode, Id. Thus, all the elements of figure 3 are run in the same mode, each task of figure 3 equally directly runnable by the operating system, and not by application software. Certainly to the extent that the Office Action relies upon the various services listed in figure 3 as disclosing the recited first and second task control blocks, Mateosian does not disclose, or even suggest, first and second memory areas in which the task control blocks are located, such that a particular task with which both task control blocks are associated can access one of the memory areas and not the other memory area.

Furthermore, although Mateosian discusses a set of first registers accessible while in a system mode (refresh register, PSAP control registers, and system mode stack register) and a second register accessible while in a normal mode (normal mode stack register), nowhere does Mateosian disclose, or even suggest, that these registers are memory areas in which are located task control blocks that contain information pertaining to a task, much less the same particular task. Furthermore, nowhere does Mateosian disclose, or even suggest, that a particular task with which both the first and second registers are associated, can access the second register, but not the first set of registers. Certainly to the extent the Office Action relies upon the underlying instructions of the software elements of figure 3 as the first and second task control blocks, nowhere does Mateosian disclose, or even suggest, that the underlying instructions are divided amongst the first and second registers. As discussed above, Mateosian states that the software elements of figure 3 are all equally run in system mode.

Thus, nowhere does Mateosian disclose, or even suggest, first and second memory areas, such that (i) a first task control block including data structures containing task information is located in the first memory area, (ii) a second such task control block is located in the second memory area, (iii) both task control blocks are associated with a particular task, and (iv) the task with which the two task control blocks are associated can directly access one of the memory areas and not the other memory area. Thus, Mateosian does not disclose, or even suggest, each limitation of claim 1.

It is therefore respectfully submitted that Mateosian does not anticipate claim 1.

With respect to claims 2 to 13, which ultimately depend from claim 1, it respectfully submitted that Mateosian does no anticipate these dependent claims for at least the same reasons set forth above with respect to the patentability of claim 1.

Claim 14 recites, inter alia, the following:

> **. . . receiving a request to create a task;**
>
> **assigning task information for the task;**
>
> **creating a first task control block for the task;**
>
> **loading the task information for the task into the first task control block;**
>
> **creating a second task control block for the task, the second task control block having a location in a memory space;**
>
> **loading an address for the location of the second task control block into the first task control block.**

As discussed above, nowhere does Mateosian disclose, or even suggest, the memory spaces in which are loaded task control blocks for a task. Although figure 1 of Mateosian illustrates the use of RAM and ROM, and although Mateosian discusses registers, Mateosian does not discuss how task control blocks for a task relate to the RAM, ROM, and registers. As discussed above, Mateosian does not disclose, or even suggest, two task control blocks for a particular task, much less an address for the location of the second task control block loaded into the first task control block. Thus, Mateosian does not disclose, or even suggest, a method that includes creating a task, creating two task control blocks for the task, loading information for the task in the first task control block, and loading an address of the second task control block into the first task control block. Thus, Mateosian does not disclose, or even suggest, each limitation of claim 14.

It is therefore respectfully submitted that Mateosian does not anticipate claim 14.

With respect to claims 15 and 16, which depend from claim 14, it is respectfully submitted that Mateosian does not anticipate these dependent claims for at least the same reasons set forth above with respect to the patentability of claim 14.

Claim 17 recites, inter alia, the following:

> **. . . receiving a context switching event;**
>
> **saving task information for a first task in a system task control block associated with the first task, the task information for the first task including a pointer to a user task control block associated with the first task . . .**

As discussed above, nowhere does Mateosian disclose, or even suggest, two task control blocks associated with a particular task, much less a first task control block that includes a pointer to a second task control block, and much less a system control block and a user control block associated with a particular task. Thus, Mateosian does not disclose, or even suggest, all the limitations of claim 17.

It is therefore respectfully submitted that Mateosian does not anticipate claim 17.

Claim 18 recites, inter alia, the following:

> **. . . receiving an interrupt request;**
>
> **saving a first number of state information values from a current task data structure for a currently executing task, the current task data structure including a pointer data structure holding a pointer to a second number of state information values . . .**

As discussed above, other than the general list of software elements of figure 3, Mateosian does not discuss the underlying tasks of the software elements of figure 3. For example, Mateosian does not discuss the underlying information or data structures of tasks. Nowhere does Mateosian disclose, or even suggest, a method in which (i) an interrupt request is received, and (ii) a first number of state information values from a task data structure is saved, where the data structure including the first number of state information values also includes a pointer to a second number of state information values. Thus, Mateosian does not disclose, or even suggest, all the limitations of claim 18.

It is therefore respectfully submitted that Mateosian does not anticipate claim 18.

With respect to claims 19 to 21, which ultimately depend from claim 18, it is respectfully submitted that Mateosian does not anticipate these dependent claims for at least the same reasons set forth above with respect to the patentability of claim 18.

III. Conclusion

In light of the foregoing, it is respectfully submitted that all pending claims 1 to 21 are in condition for allowance. Prompt reconsideration and allowance of the present application are therefore earnestly solicited.

Respectfully submitted,

Dated: 7 July, 2004 By: _____

Michelle Carniaux
Reg. No. 36,098

KENYON & KENYON
One Broadway
New York, New York 10004
(212) 425-7200

CUSTOMER NO 26646